ARHITECTURA SISTEMELOR DE CALCUL - CURS 0x00

INFORMAȚII ADMINISTRATIVE

Cristian Rusu

CUPRINS

- ~~cadre didactice~~
- ~~organizare~~
- ~~evaluare~~
- structura cursului
- obiectivele cursului
- ~~referințe bibliografice generale~~

STRUCTURA CURSULUI

- **circuite digitale**
  - teoria informației și abstractizarea digitală
  - funcții și circuite logice

- **arhitecturi de calcul** ← materia acoperită la laborator începe de aici
  - seturi de instrucțiuni
  - limbajul assembly
  - compilatoare
  - pipelining
  - ierarhia memoriei

- **organizarea calculatoarelor**
  - unitatea de procesare centrală
  - performanța calculatoarelor
  - dispozitive periferice și întreruperi
  - calcul paralel

- **potențiale subiecte moderne la curs: RISC-V, WebAssembly, TockOS, hardware pentru machine learning (GPU, TPU, etc.), etc.**

- **laborator: progamare în limbajul Assembly x86**

STRUCTURA CURSULUI

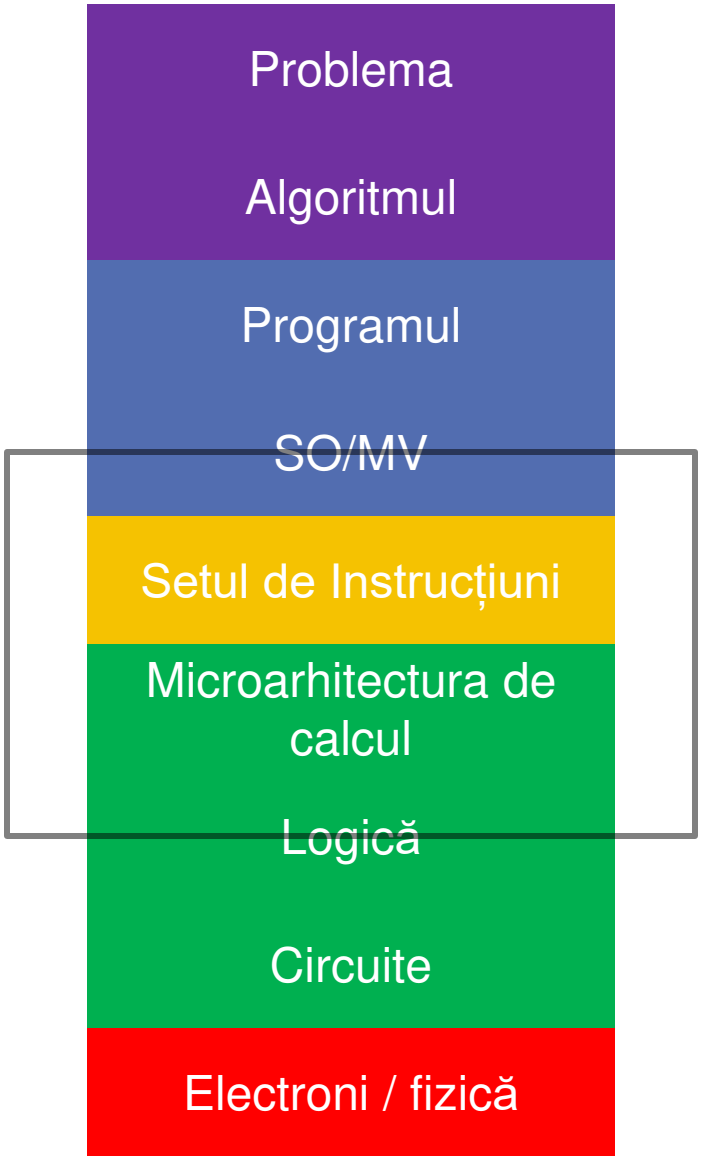
- **laborator: progamare în limbajul Assembly x86**

- **nimeni (foarte puțină lume) programează doar/complet în Assembly**

- **assembly x86 este folositor pentru:**
  - securitate informatică
    - Reverse Engineering (RE)
    - hacking
  - optimizare
    - dezvoltare jocuri
    - Machine Learning (ML/AI)
  - debugging
  - dezvoltare software *low-level*
    - dezvoltare pentru sisteme embedded
    - dezvoltare pentru sisteme de operare

.

POZIȚIONAREA CURSULUI

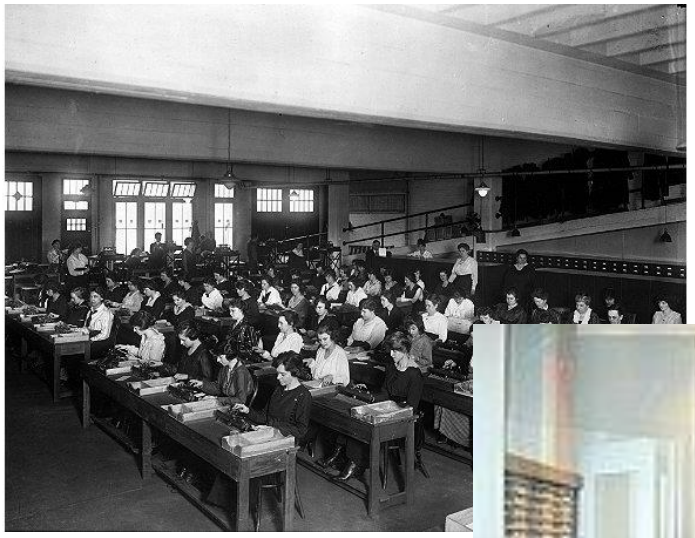


OBIECTIVELE CURSULUI

- **să înțelegeți principile arhitecturii sistemelor de calcul**
  - ce urmărim
  - ce limitări există
  - punerea în balanță a unor criterii de performanță contradictorii
- **să înțelegeți ce există ca tehnologie acum**
  - ce limitări există astăzi
  - ce execută un computer pe care îl programați voi
  - cum puteți optimiza execuția programelor
- **să înțelegeți cum să folosiți ce ați învățat în viitor**
  - design de hardware nou
- **pentru a ne asigura că lucrurile merg bine, veți avea posibilitatea să oferiți feedback (anonim binențeles) la jumătatea cursului**
- **pe parcurs, dacă sunt probleme sau neclarități vă rog sa mi le comunicați (e-mail, la clasă, fie direct sau anonim sau printr-un reprezentant, etc.)**

"The purpose of computing is insight, not numbers." (Richard Hamming)

PROGRESUL TEHNOLOGIC FĂCUT



- înainte de al doilea război mondial, 1939, USA
- putere de calcul: 50 operații / secundă

- ENIAC
- 1947 – 1955, USA
- 1000 operații / secundă

- HPE CRAY, 2021, SUA
- core-uri: 8.699.904
- 1714 peta operații / secundă

https://www.theatlantic.com/technology/archive/2013/10/computing-power-used-to-be-measured-in-kilo-girls/280633/
https://en.wikipedia.org/wiki/ENIAC
https://www.top500.org

CERINȚELE DE AZI (EXEMPLU)

- **hardware pentru machine learning (învățare automată)**
- **DeepMind și OpenAI vorbesc despre două lucruri:**
  - algoritmi noi pentru modelare și antrenare
  - “compute” (infrastructura hardware pe care rulează algoritmii)

- **AlphaZero: software bazat de ML**
  - Stockfish (software clasic bazat pe metode de căutare – în principal alpha/beta pruning)
    - 3500 ELO
  - AlphaZero vs Stockfish:
    - +155 -6 =839
  - Magnus Carlsen (campionul mondial actual)
    - < 2900 ELO
  - AlphaZero (alb) vs Stockfish (negru)

https://science.sciencemag.org/content/362/6419/1140.full , https://en.wikipedia.org/wiki/AlphaZero , https://www.youtube.com/watch?v=lFXJWPhDsSY

CERINȚELE DE AZI (EXEMPLU)

- **hardware pentru machine learning (învățare automată)**
- **DeepMind și OpenAI vorbesc despre două lucruri:**
  - algoritmi noi pentru modelare și antrenare
  - “compute” (infrastructura hardware pe care rulează algoritmii)

- **AlphaZero: software bazat de ML**
  - Stockfish (software clasic bazat pe metode de căutare – în principal alpha/beta pruning)
    - 3500 ELO
  - DeepMind spune că a antrenat acest algoritm 4 ore (învățat din self-play)
    - da, 4 ore pe platforma lor de calcul
    - un calcul rapid, aproximativ, arată că pe laptop-ul meu aceeași procedură de antrenare ar dura 30 de ani
    - costul? aproximativ 20$ milioane
  - cum scădem costul?
    - algoritmi mai eficienți
    - hardware special, dedicat

https://science.sciencemag.org/content/362/6419/1140.full , https://en.wikipedia.org/wiki/AlphaZero , https://keplerlounge.com/artificial/intelligence/2019/03/24/alpha-go-zero.html

CERINȚELE DE AZI (EXEMPLU)

- **ASC e importantă și pentru software-ul folosit de zi cu zi**
- **ce face următorul algoritm?**

```
# varianta A
for (int i = 0; i < n; ++i)
  for (int j = 0; j < n; ++j)
     for (int k = 0; k < n; ++k)
        C[i][j] += A[i][k] * B[k][j];

# varianta B
for (int j = 0; j < n; ++j)
  for (int i = 0; i < n; ++i)
     for (int k = 0; k < n; ++k)
        C[i][j] += A[i][k] * B[k][j];
```

- **înmulțește două matrice**
- **care este complexitatea numerică?**
  - $O(n^3)$
  - $2n^3$ operații
- ce face varianta B?
- cei doi algoritmi sunt echivalenți, matematic

pe sistemul meu de calcul varianta B este de 4 ori mai lentă decât varianta A
cum este posibil așa ceva?
care este diferența dintre cele două variante?
ordinea în care se execută instrucțiunile este foarte importantă

CERINȚELE DE AZI (EXEMPLU)

- **ASC e importantă și pentru software-ul folosit de zi cu zi**
- **ce face următorul algoritm?**

```
# varianta A
for (int i = 0; i < n; ++i)
  for (int j = 0; j < n; ++j)
     for (int k = 0; k < n; ++k)
        C[i][j] += A[i][k] * B[k][j];

# varianta B
for (int j = 0; j < n; ++j)
  for (int i = 0; i < n; ++i)
     for (int k = 0; k < n; ++k)
        C[i][j] += A[i][k] * B[k][j];
```

- **înmulțește două matrice**
- **care este complexitatea numerică?**
  - $O(n^3)$
  - $2n^3$ operații
- există un algoritm care calculează rezultatul *C* în $O(n^{2.8074})$
  - din păcate, pe arhitecturile de calcul moderne, acest algoritm este mai lent decât metoda clasică (o formă de varianta A)

concluzii:

- complexitatea numerică este foarte importantă, dar nu este totul
- ce calculăm matematic și ce putem executa sunt două lucruri diferite
- câteodată progresul în lumea reală este surprinzător și complet neevident

https://en.wikipedia.org/wiki/Strassen_algorithm